ARL-TR-8059 ● July 2017

US Army Research Laboratory

Using Cognitive Control in Software-Defined Networking for Port Scan Detection

by Vinod K Mishra and Tamarick Kindrick

Approved for public release; distribution is unlimited.

NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.

ARL-TR-8059 ● July 2017

US Army Research Laboratory

Using Cognitive Control in Software-Defined Networking for Port Scan Detection

by Vinod K Mishra
Computational and Information Sciences Directorate, ARL

by Tamarick Kindrick
North Carolina A&T State University, Greensboro, NC

Approved for public release; distribution is unlimited.

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704-0188*

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing the burden, to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.
PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.

1. REPORT DATE *(DD-MM-YYYY)* July 2017	**2. REPORT TYPE** Technical Report	**3. DATES COVERED (From - To)** 15 June–31 July 2016

4. TITLE AND SUBTITLE Using Cognitive Control in Software-Defined Networking for Port Scan Detection	**5a. CONTRACT NUMBER**
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Vinod K Mishra and Tamarick Kindrick	**5d. PROJECT NUMBER**
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) US Army Research Laboratory ATTN: RDRL-CIH-S Aberdeen Proving Ground, MD 21005-5067	**8. PERFORMING ORGANIZATION REPORT NUMBER** ARL-TR-8059
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	**10. SPONSOR/MONITOR'S ACRONYM(S)**
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT
Software-defined networking (SDN) is a new paradigm of networking in which the control plane is separated from the data plane. OpenFlow is the most prevalent protocol for communication between the SDN Controller (SDNC) and the packet forwarding device. In the present work, we studied the incorporation of the cognitive elements in the SDNC policy engine. The results were analyzed through modeling and simulation and applied to the chosen scenario of port scanning.

15. SUBJECT TERMS
port scan detection, software-defined networking, SDN, cognitive controller, SDNC

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** Vinod K Mishra
a. REPORT Unclassified	**b. ABSTRACT** Unclassified	**c. THIS PAGE** Unclassified	UU	16	**19b. TELEPHONE NUMBER (Include area code)** 410-278-0114

Standard Form 298 (Rev. 8/98)
Prescribed by ANSI Std. Z39.18

Contents

Approved for public release; distribution is unlimited.

List of Figures

Approved for public release; distribution is unlimited.

1. Introduction

In traditional networks, the network devices perform functions of control and data forwarding in the same node (Fig. 1).

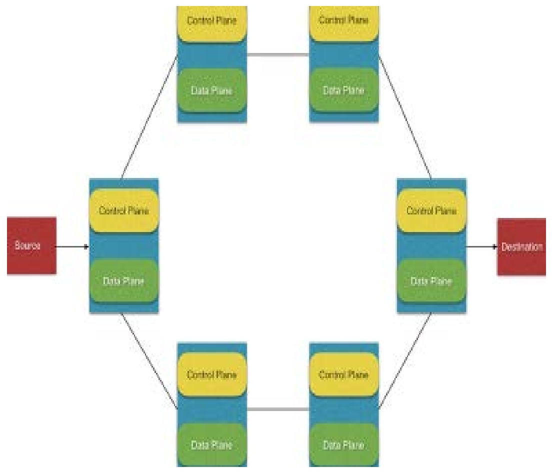


Fig. 1 Schematic of traditional network

This intertwining of both functions leads to static and inflexible networks. The Software Defined Networking (SDN)[1] is a new paradigm of networking in which these functions are separated. The control functions are removed from the devices and put in a centralized SDN Controller (SDNC).

An SDN network has the following functional elements:

- Data Plane consists of forwarding devices (FDs; e.g., switches that send incoming flows to their destinations according to rules defined in the flow tables).
- Control Plane (CP) makes rules for forwarding and modifying flows and ensures that the FDs comply with them. These functions are exercised by a central network element, or SDNC, which communicates with southbound FDs using a protocol. Currently OpenFlow[2] is the most widely used protocol for this purpose.
- Application Plane (AP) contains many services like security, load balancing, routing, firewalls, and the like. The CP communicates with AP using North Bound Interfaces for which a standardized protocol is not available at this time.

The Management Plane handles tasks like setting up of network and its configuration parameters. It is not programmable and is isolated.

Approved for public release; distribution is unlimited.

2. Cognitive Control

2.1 Need for Cognitive Control

The current SDNC is a policy-driven entity. The network devices operate on packets according to whether the incoming packet header pattern matches with existing entries in the flow table.[1,2] In the event of no match, the packet is sent to the SDNC, which decides what to do with it based on the policies. This approach has severe limitations, as its applicability depends on the existence of policies to cover a vast number of possible deviations from the normal or baseline situation. As an example, the SDNC cannot make appropriate decisions when confronted with zero-day attack packets. The cognitive intelligence applied to SDNC can alleviate most of these issues.

Earlier works on detecting the traffic anomaly due to port scan attack in an SDN network have focused more on architecture and flow management methods.[3–6] In this work, we seek to find a signature of the port scan attack by applying a simple cognitive algorithm on the control plane.

2.2 Elements of Cognitive Control

The cognitive control has the following 3 elements, as illustrated in Fig. 2:

- Network devices as sensors and observers of the networking environment
- A store of prior knowledge consisting of policy decisions based on earlier interactions of network with its environment
- Cognitive Controller (CC) as a planning and execution module taking appropriate actions when presented with newly sensed and observed networking data (Fig. 3)

The second step can be implemented by using the principles of neural networking.

Approved for public release; distribution is unlimited.

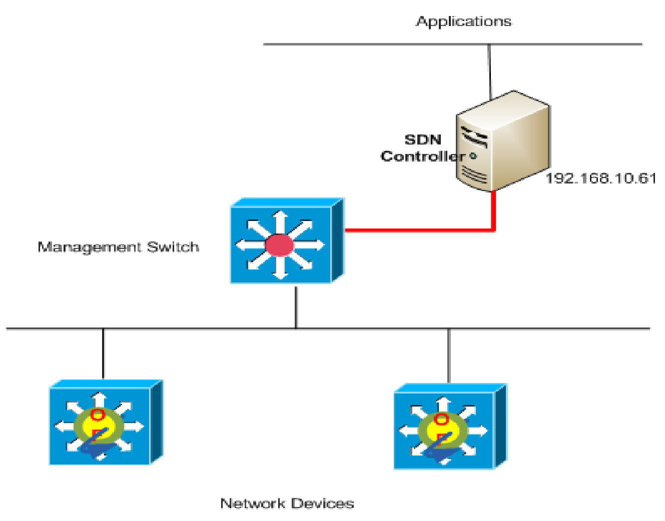


Fig. 2 Schematic of software-defined network

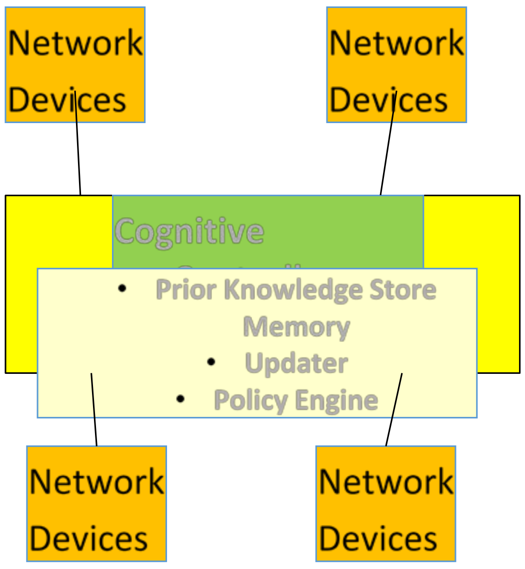


Fig. 3 Elements of cognitive controller

3. Simplified Cognitive Control Model for Port Scan Detection

3.1 Port Scan Signature

Let

T = a chosen time-interval,

N = a chosen number, and

$\nu(T)$ = the number of consecutive packets received during T.

Approved for public release; distribution is unlimited.

The source and destination information inside the arriving packets is assumed to be random. The signature of a hacker executing a port scan attack will be the probing of different port numbers to find the weaknesses in the network.

In Eq. 1, $n(T)$ is the number of consecutive packets having *same source* but *different port numbers* arriving in time interval T. Then the signature of the port scan attack is given by the following relation:

$$n(T) \geq N. \tag{1}$$

3.2 Logical Steps in the Port Scan Detection Process

Our simplified CC model for detecting port scan attack has the following entities:

- 2 OpenFlow switches (Sw-1 and Sw-2) that contain flow tables (FTs) with baseline rules
- The CC is the module of the general SDNC with intelligence to recognize port scan attacks. It has access to a memory buffer for storage and contains a Time Counter cycling between 0 and T. It also stores a chosen threshold number, N, defined earlier.

The CC examines the packet flows according to the following logic.

At the start, the switch Sw-1 receives a packet from the packet source Src-host.

1) Sw-1 matches the header attributes of a packet against the rules in its FT and enforces the rule for a matching entry.
2) Sw-1 sends a nonmatching packet to the CC.
    a. The CC reads the header attributes source (s_n), destination (d_n), and port number (p_n) as a 3-tuple at the related time instant, t_n. It compares them with entries in an internal table of black listed sources.
    b. For matching s_n, CC instructs both Sw-1 and Sw-2 to drop all incoming packets from that source.
    c. For nonmatching s_n, CC sends the packet to a decision block.
3) CC compares the current source value s_n with earlier instant source value s_{n-1}.
    a. For $s_n \neq s_{n-1}$ (different sources in consecutive packets), CC instructs both Sw-1 and Sw-2 to add a new rule corresponding to this packet to their FTs.

Approved for public release; distribution is unlimited.

   b. For $s_n = s_{n-1}$ (same source in consecutive packets), CC sends the packet to a new decision block.

4) CC compares the current port value p_n with earlier instant port value p_{n-1}.

   a. For $p_n = p_{n-1}$ (same port in consecutive packets), CC instructs Sw-1 and Sw-2 to add a new rule corresponding to this packet to their FTs.
   b. For $p_n \neq p_{n-1}$ (different ports in consecutive packets), CC sends the packet to a new decision block.

5) CC compares the packet arrival sequence number *n* (an integer) with a preset integer N.

   a. For $n \leq N$ (the packet counter less than or equal to the preset integer), CC lets the packet go to its destination.
   b. For $n > N$ (the packet counter greater than the preset integer), CC determines the arrival of the packet as signature of a port scan attack. CC immediately blocks the source, puts it in the black list, and sends an alert to the management.

3.3 Model Used in Current Approach

In this report, we have used the logical steps described in Section 3.2 as guidance for the current investigations as given in Fig. 4.

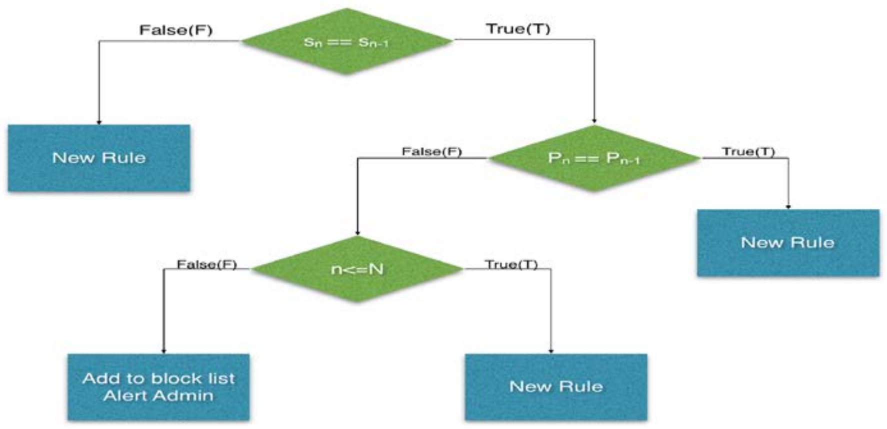
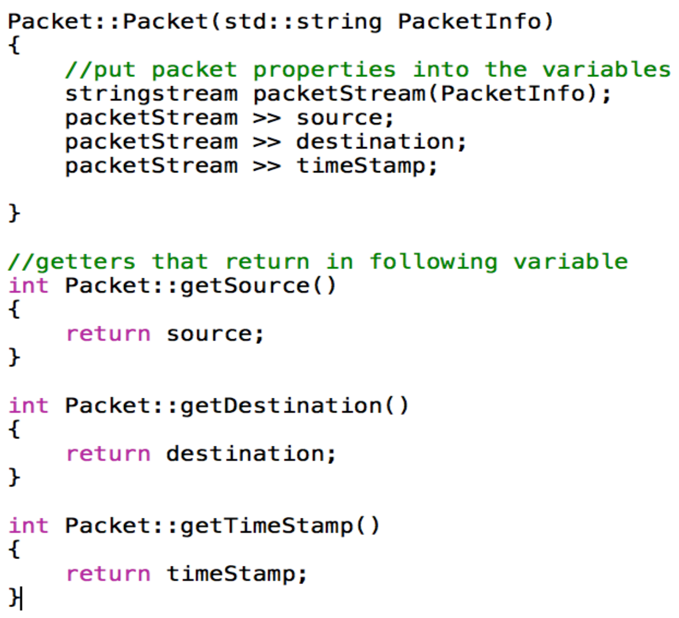


Fig. 4 Decision blocks for cognitive control

4. CC Module

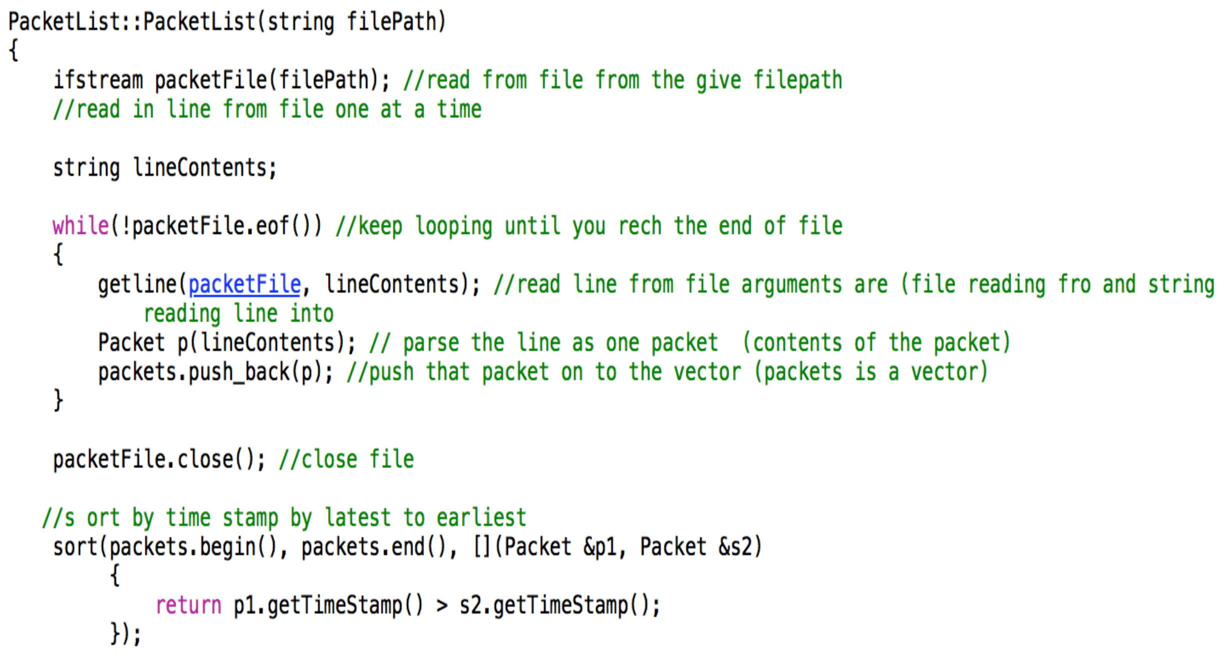
The steps shown in Fig. 4 were coded in C++, and the raw packets were taken from a CAIDA archive[7] of real-time network data. The operational logic coded in C++ is shown in the following code.

Approved for public release; distribution is unlimited.



```
Packet::Packet(std::string PacketInfo)
{
    //put packet properties into the variables
    stringstream packetStream(PacketInfo);
    packetStream >> source;
    packetStream >> destination;
    packetStream >> timeStamp;

}

//getters that return in following variable
int Packet::getSource()
{
    return source;
}

int Packet::getDestination()
{
    return destination;
}

int Packet::getTimeStamp()
{
    return timeStamp;
}
```

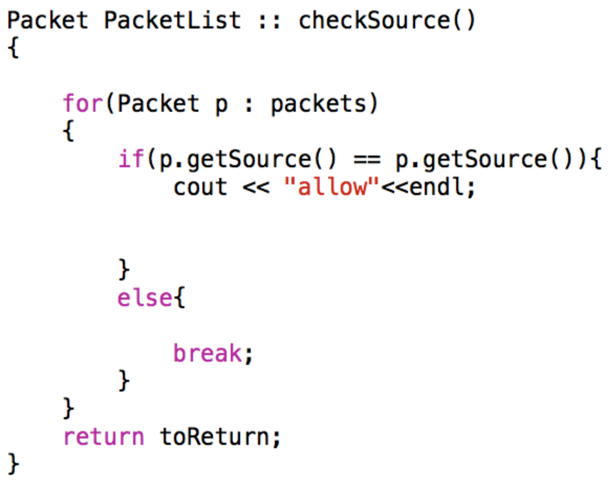
The following code snippet describes the Packet object and its attributes.



```
PacketList::PacketList(string filePath)
{
    ifstream packetFile(filePath); //read from file from the give filepath
    //read in line from file one at a time

    string lineContents;

    while(!packetFile.eof()) //keep looping until you rech the end of file
    {
        getline(packetFile, lineContents); //read line from file arguments are (file reading fro and string
            reading line into
        Packet p(lineContents); // parse the line as one packet  (contents of the packet)
        packets.push_back(p); //push that packet on to the vector (packets is a vector)
    }

    packetFile.close(); //close file

   //s ort by time stamp by latest to earliest
    sort(packets.begin(), packets.end(), [](Packet &p1, Packet &s2)
        {
            return p1.getTimeStamp() > s2.getTimeStamp();
        });
```

The following code gives a sample of a logic test. If the sources are the same, this will be allowed and will print that this is okay; if not, it will break out of this check.

Approved for public release; distribution is unlimited.



```
Packet PacketList :: checkSource()
{

    for(Packet p : packets)
    {
        if(p.getSource() == p.getSource()){
            cout << "allow"<<endl;


        }
        else{

            break;
        }
    }
    return toReturn;
}
```

5. Conclusions

In the current work, we showed that a simple algorithm was able to identify the traffic anomaly due to a port scan attack This is an example of cognitive intelligence, which can be added to the SDNC as part of its computing engine. In the future we want to investigate extensions of this approach to a larger network with more-complex topology. We also want to investigate other algorithms to detect a denial of service attack in conjunction with port scan attack. This algorithm can be implemented as a module in any SDNC (e.g., NOX or OpenDayLight). We will also test this approach with socket programming with real network traffic on the GENI/OpenFlow Platform.

Approved for public release; distribution is unlimited.

6. References

1. Bosshart P, Gibb G, Kim H, Varghese G, McKeown N, Izzard M, Mujica F, Horowitz M. Forwarding metamorphosis: fast programmable match-action processing in hardware for SDN. Proceedings of ACM SIGCOMM ‘13; 2013 [accessed 2016 July]. http://yuba.stanford.edu/~grg/docs/sdn-chip-sigcomm -2013.pdf.

2. Jayawardena M, Chijioke JK, Oluwayemisi O. Open Flow v1.4 and Open VSwitch 2.4.0; 2015 Dec. http://www.ResearchGate.net.

3. Mehdi SA, Khalid J, Khayam SA. Revisiting traffic anomaly detection using software defined networking; 2011 [accessed 2016 July]. http://pages.cs.wisc .edu/~junaid/papers/raid2011.pdf.

4. Shirali-Shahreza, Ganjali Y. Efficient implementation of security applications in openflow controller with FleXam; 2013 [accessed 2016 July]. https://pdfs.semanticscholar.org/a8ec/b1b996ad62fc18c8a51f53fefc3192310 7b3.pdf.

5. Cabaj K, Wytrebowicz J, Kuklinski S, Radziszewski P, Dinh KT. SDN architecture impact on network security. Proceedings of the Federated Conference on Computer Science and Information Systems; 2014. p. 143–148

6. Shin S, Yegneswaran V, Porras P, Gu G. AVANT-GUARD: scalable and vigilant switch flow management in software-defined networks. Proceedings of the ACM SIGSAC Conference on computer and Communications Security; 2013. New York (NY): Association for Computing Machinery (ACM); 2013. doi.10.1145/2508859.2516684.

7. Network packet header data. San Diego (CA): Center for Applied Internet Data Analysis (CAIDA); 2016 July.

Approved for public release; distribution is unlimited.

List of Symbols, Abbreviations, and Acronyms

AP	Application Plane
CC	Cognitive Controller
CP	Control Plane
d_n	destination
FD	forwarding device
FT	flow table
p_n	port number
SDN	Software-Defined Networking
s_n	source
SDNC	Software-Defined Networking Controller
t_n	time instant

Approved for public release; distribution is unlimited.

1 (PDF)	DEFENSE TECHNICAL INFORMATION CTR DTIC OCA
2 (PDF)	DIRECTOR US ARMY RESEARCH LAB RDRL CIO L IMAL HRA MAIL & RECORDS MGMT
1 (PDF)	GOVT PRINTG OFC A MALHOTRA
1 (PDF)	DIR USARL RDRL CIH S V K MISHRA

Approved for public release; distribution is unlimited.